Automatic Mesh Generation, Geometric Modeling, and Visualization Tool Development Efforts at the Army HPC Research Center / NetworkCS

Andrew A. Johnson

Army HPC Research Center,
Network Computing Services, Inc.
1200 Washington Ave. S.
Minneapolis, MN 55415
Email: ajohn@networkcs.com

Abstract

In this paper I briefly describe some of the ongoing research and development efforts into fast and efficient numerical simulation tools applied to computational fluid dynamics (CFD) applications. While these tools have been mainly designed for CFD applications, it is expected that these tools can be applied to other research areas such as computational solid mechanics. Specifically, this work has involved the development of geometric modeling software, 3D automatic mesh generation tools, and a parallel /distributed interactive data visualizer.

Introduction

In this paper I describe some of the research and software development efforts in numerical simulation tools at the Army High Performance Computing Research Center (AHPCRC) / Network Computing Services, Inc. These tools include geometric modeling and automatic mesh generation in 3D, and large-scale (parallel) unstructured data visualization based on a client-server approach. All development efforts are geared towards large data sets, applications involving complex geometry in 3D, and interfaces which can be run from users' desktop computers.

The automatic mesh generation and modeling tool is composed of three parts. The first part involves geometric modelers based on either Bezier or NURBS surfaces. With this tool, the actual geometry and computational domain is defined. The second tool is used to automatically discretize the geometric model into a surface mesh composed of triangular elements. The control of the refinement level within this surface mesh is quite easy. The third stage is the actual 3D automatic mesh generation where the volume of the domain is discretized into tetrahedrons. These tools are based on Delaunay methods and incorporate face-swapping techniques. Typical mesh sizes range between 1 and 2 million elements, but the system has been tested up to around 5 million. Although began as a research project, these tools have proven to be quite easy-to-use, fairly robust, and fast [1,2].

Our research and development efforts into fast and efficient visualization techniques has lead to a system based on a client-server approach. In many simulations, the amount of data that is generated is very large and resides on a remote parallel computer that generated that data, such as the Cray T3E at the Army HPC Research Center. It is difficult and impractical in many cases to transfer all this data to a local workstation and visualize it effectively on that workstation. To

avoid this, our tool is written in a framework where the processing of the large 3D data set is handled by a remote server program running in parallel, and the visualization and interaction is handled by the client program running on the user's desktop. The two programs communicate through standard Internet protocols. By writing our visualization tool in this way, large 3D data sets can be visualized quickly and efficiently from a user's desktop.

These simulation tools have been applied to a wide variety of applications such as those involving airdrop systems, flows in waterways and hydrodynamics, and other computational fluid dynamics related problems [1-3].

Automatic Mesh Generation and Modeling

Generating a mesh for numerical simulation involving any sort of complex geometry can be a very time consuming and tedious task. With more traditional structured mesh generation methods, many weeks or months can be spent generating a mesh around a complex shape. Because of this difficulty in creating meshes for complex geometry, and the ever increasing necessity for fast turn-around times in numerical simulation, automatic mesh generation methods have become increasingly popular. Our work on automatic mesh generation methods has been ongoing for several years, and this tool has been applied to a wide variety of applications [1-3]. Our development work in this area has been driven by our necessity to solve very realistic and complex applications, and we found that automatic mesh generation was the only practical option which allows us to discretize problem domains very quickly. Out of this development effort has come a very useful and widely applicable mesh generation tool set.

Our mesh generation tool actually consists of three components. The first is an interactive computer aided design (CAD) program which allows researchers to build the 3D geometry being studied. The second component is an automatic surface mesh generator where the geometric model is automatically discretized into a surface mesh composed of triangular elements. The final component is the actual 3D automatic mesh generator which discretizes the problem domain into tetrahedral (i.e. four nodded) elements.

Geometric Modeling

Interactive geometric modeling has been available to engineers for quite some time. Our development of geometric modeling software was mainy due to necessity, but also serves as the front-end to our automatic mesh generation tools where the actual geometry under consideration is defined. We view this tool as serving two purposes. While our geometric modeling tool is not in the class as a ProEngineer or other professional CAD package, simple to moderately complex geometry can be created interactively without too much difficulty. If an engineer or scientist wants to generate a mesh for some shape they are studying, they may not have the resources, expertise, or time to use a professional CAD program, so we use our simple geometric modeler to accomplish this. Our tool is called ModelGui and is fairly easy-to-use, and geometric models are build from either triangular and/or quadrilateral Bezier surfaces [4]. Model surfaces are build on top of curves, which are build on top of points in 3D space, and the tool has several helpful features when interactively building a model. A screen shot of ModelGui is shown in Figure 1.

The second purpose of our modeling tools is to import and clean-up third party models that were generated with some other professional modeling package. Third party models (i.e. models not generated with our software) are often not in a state where a 3D mesh can be created right away. Often times a model must be cleaned-up where all seams are filled, overlaping surfaces are eliminated, and in most cases, the external boundaries of the computational domain must be defined. To support the importing and clean-up of complex third party models, we are finishing development of a new modeling program called Anemone. The main developer of Anemone is Steve Demlow of the AHPCRC / Network Computing Services, Inc. This program can import NURBS-based models in the IGES format, and has a very professional 3D visualization interface (see Figure 2 for a preliminary screen shot of Anemone). This modeler supports NURBS surfaces and can also handle trimmed-NURBS surfaces in most cases. We are moving support from our old geometric modeler ModelGui to this new and more professional package since Anemone also has all of the tools necessary to build and modify geometric models from scratch. Both of our geometric modeling tools (ModelGui and Anemone) have build-in features to support automatic mesh generation requirements such as the ability to specify desired mesh refinement information at the corners of all surfaces. The automatic mesh generation tools (described next) will use this refinement information to determine the element sizes in the final 3D mesh.

Automatic Surface Mesh Generation

After a "valid" geometric model is built, the next step is to discretize the model into a surface mesh composed of triangular surface elements. Due to numerical simulation requirements, it is necessary for the surface mesh generator to create triangles of very high quality (i.e. as close to an equilateral triangle as possible) and allow only smooth variations in element size across the model. Secondly, it is important to be able to easily control the size of the triangles (i.e. the refinement level) at various locations throughout the model. The size of these triangles will have a direct result on the size of the final 3D mesh, which will both determine the size of the numerical simulation (i.e. computer power and memory required) and its accuracy.

Our automatic surface mesh generation tool is called DimensionSG and has both of these features. It uses a method very similar to automatic 2D mesh generation by Delaunay methods [5]. This method generates both high-quality triangles, and it is also very easy to control mesh refinement. As stated before, refinement parameters (i.e. desired edge-length) were set in the geometric model itself and will control the size of the triangles near those locations. Using this refinement information, each model surface (Bezier surface or NURBS) is discretized within its parametric u-v space with a traditional 2D Delaunay method. The modification we impose on this method in order to discretize an actual 3D surface is that all decisions needed in the Delaunay method about how to configure the elements (i.e. we use a face-swapping technique) are made on the parametric surface mapped into 3D space. By doing this, even though the mesh is generated in the parametric 2D u-v space, the shape of the elements are going to be optimal on the actual 3D model surface. By design, the discretization of each surface is independent on the underlying parametric space's transformations, even though the mesh is actually constructed on this parametric 2D surface.

Our automatic surface mesh generator can also discretize trimmed-NURBS surfaces. In a trimmed NURBS surface, a portion of the surface is "cut-out" or trimmed by other curves which are drawn within the 2D parametric u-v space. The fact that our method is based on a Delaunay one makes this possible since the u-v space does not have to be a square, which is the traditional shape of a parametric u-v domain, but can be arbitrary and represent any complex shape created by the trimming curves.

Our surface mesh generation tool also has features to detect open seams in the model geometry and any topological inconsistencies that may exist. The program will make sure that the discretization of each surface will match the discretization of its neighboring surfaces along common edges.

Our automatic surface mesh generation program DimensionSG is currently console-based since no user interaction is required. Generated surface meshes can be interactively visualized with a simple surface mesh visualization program we have developed. It is possible that we may integrate the surface mesh generation program with our interactive geometric modeler Anemone in future versions. See Figures 3 and 4 for screen shots of sample surface meshes generated with our software.

Automatic Volume Mesh Generation

The surface mesh composed of triangular elements generated in the previous stage becomes the input to our actual 3D automatic mesh generator which discretizes the volume into tetrahedral elements. The size of the tetrahedral elements that are generated is dependent upon the size of the surface triangles in the inputted surface mesh. For example, if the left side of the model is defined by large triangles, and the right side of the model is defined by small triangles, the tetrahedral elemens inbetween these two surfaces will vary linearly from large elements to small elements across that space. Because of this feature of the mesh generator, we allow users to define other surfaces within the interior of the model which are not actual physical boundaries but only exist to specify a desired mesh refinement level at various locations. We often use this feature to create what we call refinement boxes or refinement zones in order to concentrate smaller elements in locations where high-refinement is needed such as near boundary layers and in wakes.

The method to discretize the volume is based again on Delaunay methods and incorporates face-swapping techniques [1,2]. We chose to use a Delaunay method because that method can handle truly arbitrary and complex geometry, is fast, and generates fairly good quality tetrahedra. Our automatic mesh generator guarantees that the triangles of the inputted surface mesh will match exactly with the surface of the final 3D mesh composed of tetrahedral elements. This in not always the case with Delaunay methods because the method really only deals with points (i.e. nodes) and does not take into account how the nodes on the inputted surface mesh were originally connected together. Also, traditional Delaunay methods sometimes suffer from bad element quality in some cases (i.e. the well known sliver elements that can be generated with the method) so we incorporate methods to improve element quality (i.e. aspect ratio) as much as possible in the final stage of mesh generation.

Our automatic mesh generation program is called DimensionMG and can generate meshes in fairly short periods of time using simple workstations. Common times for mesh generation are between 2-4 minutes per 1 million elements. Common mesh sizes are between ½ to 2 million elements, but the program has been tested up to 4-5 million elements. The typical mode of operation is console based since not much information is required from the user during mesh generation (i.e. it is an automatic method). We have added a simple graphical user interface (GUI) to the program which is used to facilitate generation of meshes on remote servers in a client-server mode of operation. Typically, meshes are created on the user's local workstation, but the program supports this client-server interface where the mesh can actually be created on a fast/large-memory server at a remote site, and the generation is controlled form a user's desktop system. This has been helpful to us in simulations involving multi-platform computing and remeshing techniques where meshes are created multiple times during a numerical simulation.

Visualization of 3D meshes is sometimes difficult, but a cross-section of a mesh generated with DimensionMG can be seen in Figure 5.

Data Visualization Tools

Storing, transferring, visualizing, and interpreting the large data sets generated by today's parallel computers is becoming an ever increasing challenge. A typical data set is composed of an unstructured mesh (nodal coordinates and element connectivity) and solution data at the mesh's nodes and/or elements. In traditional data visualization schemes, the data that was generated is transferred to a user's workstation and stored locally. This local data set is then loaded into the workstation's memory which then processes the data set where visualization components such as boundaries, cross-sections, iso-surfaces, and streamlines are created and displayed by the workstation. In cases where the data set is large, it can be difficult to find disk storage on a local workstation, and transfer times across the Internet can become long. After the data is sent to the workstation, that workstation may not be powerful enough to load the data set into memory and process it quickly and effectively for visualization.

To overcome these bottlenecks, and to fit nicely within the Army HPC Research Center's computing framework, we have built a data visualization tool built upon a client-server model. The AHPCRC is set up with a central computing site which houses a large computing resource (namely a 1088 processor Cray T3E), and there are researchers located at many remote sites all over the country which access this central computing resource from their desktop systems. A client-server model works effectively within this framework (see Figure 6).

The data visualization tool is called the Presto Visualizer and is actually built from two components. The server program is designed to run in parallel on a large parallel supercomputer, and it is responsible for loading and processing the large data sets which were generated on that computer. The second program is the client and this runs on the user's workstation. It is responsible for handling the user interaction, and displays all of the visualization constructs (i.e. boundaries, cross-sections, iso-surfaces, etc.) requested by the user. The two programs communicate through standard Internet protocols. By visualizing a data set in this way, the data set never needs to leave the computer where it was generated.

The key to the effectiveness of this client-server model is that even though the 3D data set can be very large, the items that are created for visualization are only surfaces (boundaries, cross-sections, etc.). The amount of data needed store a surface is an order-of-magnitude smaller than the original 3D data set, so creating these surfaces and sending them across the Internet to a user's workstation is a manageable task. The user's workstation does not have to deal with the original, large 3D data set, but only deals with the surfaces that are sent to it by the server program.

Currently, the Presto Visualizer can only process a single data set (i.e. a single time-step), but support for multiple data sets (i.e. a sequence of time-steps) and animation is being added.

Geometry Server

The server part of this visualization tool is called the Geometry_Server. It is written in a distributed-memory parallel framework using the MPI library. The Geometry_Server is responsible for loading, partitioning, and distributing the unstructured mesh onto the parallel processors, and extracting any visualization constructs (i.e. surfaces) that are requested by the user. Currently, 3D tetrahedral or hexahedral data sets are supported with data (i.e. variables such as velocity, pressure, temperature, etc.) located at the nodes and/or elements. Both scalar and vector data is supported. Mesh partitioning is accomplished with the Metis library [6]. Once the mesh and data is distributed on the processors, various visualization constructs can be extracted such as boundaries, cross-sections, iso-surfaces, and streamlines (i.e. particle traces). This "extraction" of the visualization constructs is accomplished in parallel, and these surfaces are also stored in a distributed manner amongst the parallel processors. The first processor (Processor 0) actually serves as a "manager" within this parallel computing framework and does not get a portion of the 3D data set as the others do. Processor 0 is responsible for communicating with the client program on the remote workstation, coordinating the worker processors, assembling the visualization surfaces generated on all of the workers, sending the visualization surface to the client, and handles all disk I/O.

As stated before, the Geometry_Server is written with the MPI library and is therefore portable to any architecture which supports MPI such as the Cray T3E, IBM SP, and workstation clusters. Typically, we run the Geometry_Server using between 5 and 8 processors, and some modest speed-ups are gained by using more processors. The main effect of using more processors is that there is more memory available to store and process the large 3D data set. If larger data sets are visualized, we simply use more processors and gain access to a larger pool of processor memory. The upper limit on the size of the data set (i.e. size of the mesh) that can be processed by the Geometry_Server has not been fully tested.

Visualization Client

The client part of our data visualization tool is called PrestoVis and runs on the user's local desktop computer. The client is responsible for all user interaction, communicates with the Geometry_Server running on the remote parallel computer, and displays all visualization surface geometry sent to it by the server program. To simplify development and to keep PrestoVis very portable, it is written in C, uses Tcl/Tk for the user interaction components (buttons, menus,

etc.), and uses the OpenGL library to display and manipulate the 3D geometry. It has so far been ported to the Linux, SGI, and Windows platforms.

We have kept the user interface fairly simple in order to accommodate fast and easy set-up, and visualization. Components are built-in to the interface to connect to the server program, and get through any secure firewalls which may surround such systems. Once a connection is made to the server program, all interaction is through this interface and it behaves as any normal program.

Screen-shots of the visualization tool are shown in Figures 7 through 9.

Internet Considerations

The main bottleneck to effective visualization in this client-server mode is the speed of the network connection between the two systems. We use a very basic scheme based on Sockets to facilitate the client-server communication. Of course, a fast connection is best, but we have found that moderate network speed are quite effective. We have visualized data sets where the client and server computers are located a thousand miles or more from each other, and transfer times were quite within acceptable limits. We have also accomplished visualization from user's home systems across simple DSL connections, and it has also proven to be very effective.

We are currently looking into user authentication, accounting, and automatic server start-up based on Kerberos. Currently, users must log onto the remote system and start-up the server "by-hand", but we want to add functionality to the Presto Visualizer where this is not needed, and the server starts automatically if the client is run from a Kerberized workstation.

Summary

In conclusion, the Army HPC Research Center / Network Computing Services, Inc. has an active program in building numerical simulation tools for large-scale flow simulation based on unstructured mesh technology. These tools include interactive geometric modelers, automatic surface and volume mesh generation, and a data visualization tool build in a client-server framework which facilitates visualization of large data sets residing on remote parallel servers. Although began as research codes, these tools have since proven to be quite effective and can be applied to many types of applications.

Acknowledgement

This work was sponsored in part by the Army High Performance Computing Research Center under the auspices of the Department of the Army, Army Research Laboratory contract number DAAH04-95-C-0008. The content does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

References

[1] A.A. Johnson and T.E. Tezduyar, "Parallel Computation of Incompressible Flows with Complex Geometry", *International Journal for Numerical Methods in Fluids*, **24** (1997) 1321-1340.

[2] A. Johnson and T. Tezduyar, "Advanced Mesh Generation and Update Methods for 3D Flow Simulations", *Computational Mechanics*, **23** (1999) 130-143.

[3] T. Tezduyar, S. Aliabadi, M. Behr, A. Johnson, V. Kalro and M. Litke, "Flow Simulation and High Performance Computing", *Computational Mechanics*, **18** (1996) 397-412.

[4] G. Farin, *Curves and Surfaces for Computer Aided Geometric Design, A Practical Guide*, Academic Press, 1993.

[5] P.L. George, *Automatic Mesh Generation, Application to Finite Element Methods*, John Wiley & Sons, 1991.

[6] G. Karypis and V. Kumar, "Parallel Multilevel k-Way Partitioning Scheme for Irregular Graphs", *SIAM Review*, **41(2)** (1999) 278-300.

Figures

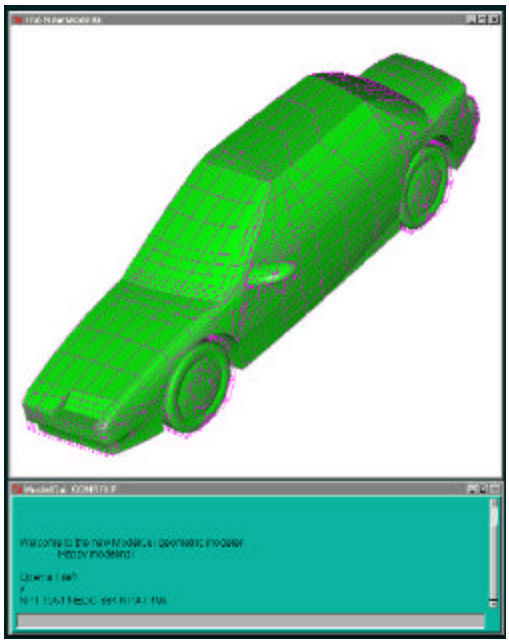

Figure 1. Model of an automobile loaded within the ModelGui geometric modeler.

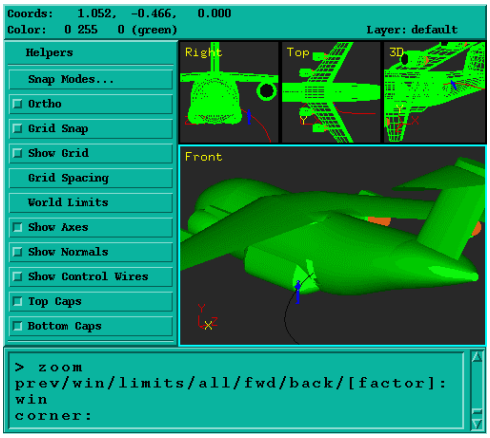

Figure 1. Model of an aircraft loaded within a preliminary version of the Anemone geometric modeler.

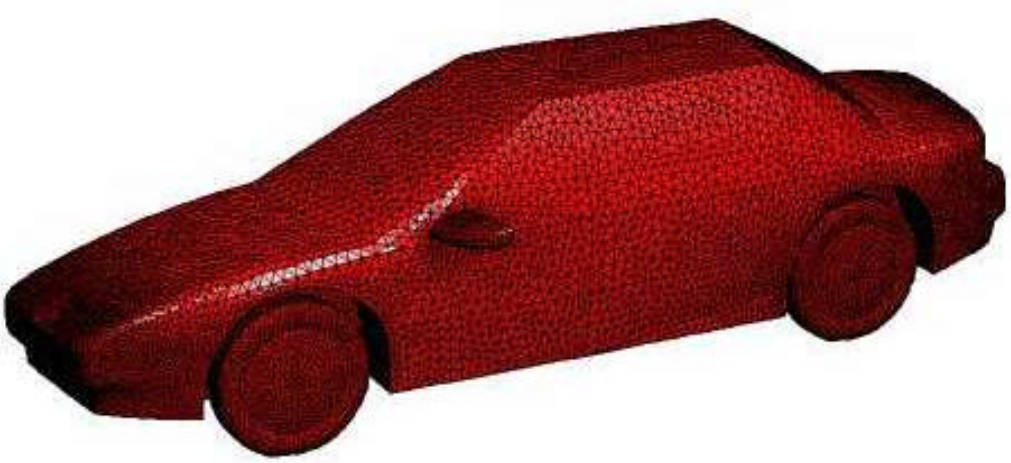

Figure 3. Triangular element surface mesh of an automobile model.

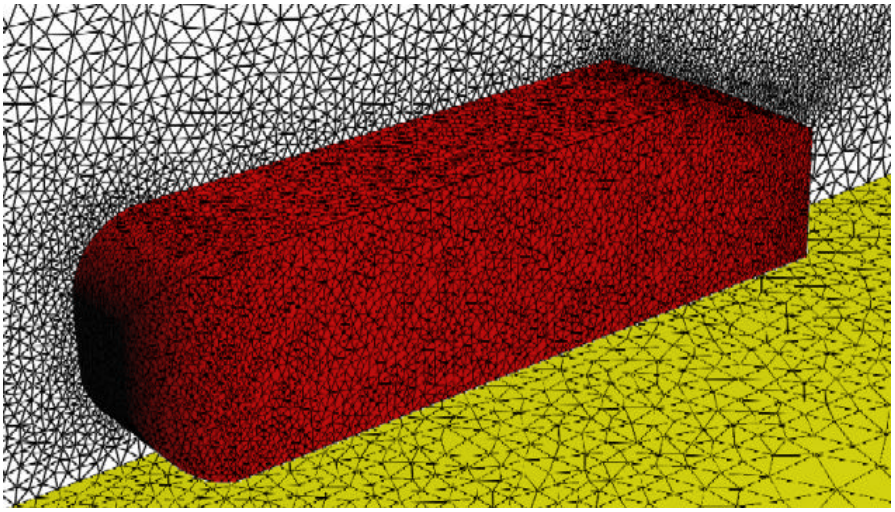

Figure 4. Triangular element surface mesh of a generic shape.

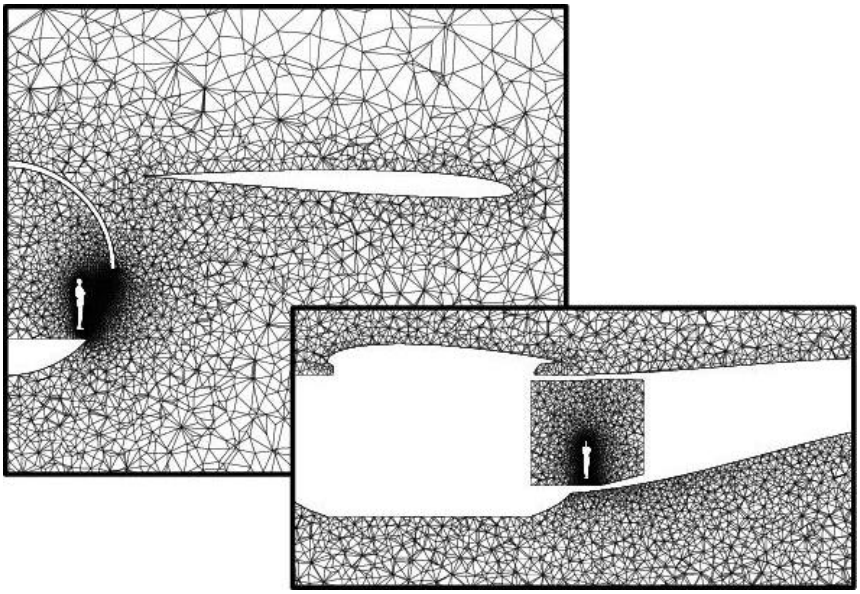

Figure 5. Cross-sections of a 3D mesh composed of tetrahedral elements.

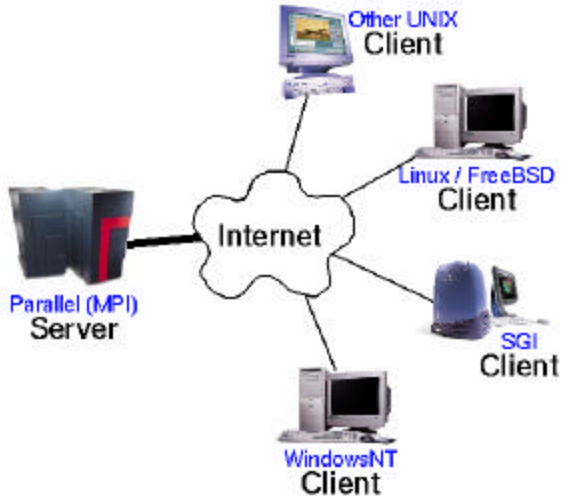

Figure 6. Schematic of a client-server computing environment.

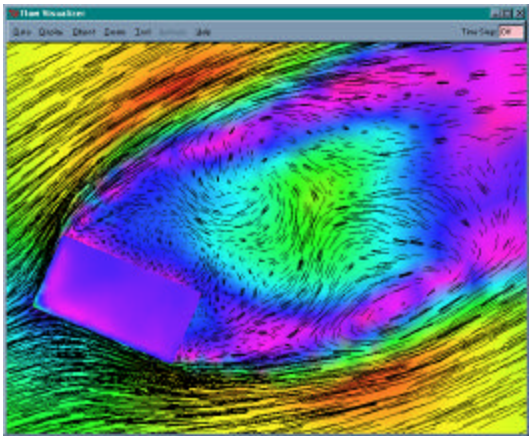

Figure 7. Screen-shot of the Presto Visualizer showing velocity vectors and flow speed at a cross-section.

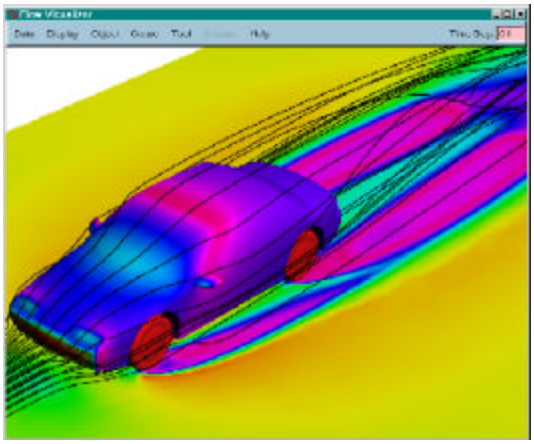

Figure 8. Screen-shot of the Presto Visualizer showing variables on boundaries and streamlines.

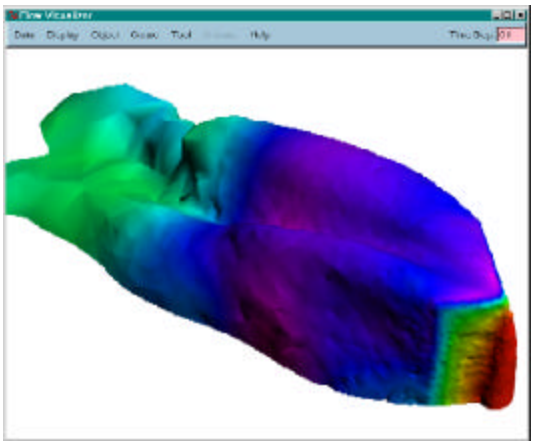

Figure 9. Screen-shot of the Presto Visualizer showing an iso-surface colored with a scalar variable.